整数行列のFrobenius標準形のモジュラー計算法 (数式処理における理論と応用の研究)

AUTHOR(S):

森継, 修一; 栗山, 和子

CITATION:

森継, 修一 ...[et al]. 整数行列のFrobenius標準形のモジュラー計算法 (数式処理における理論と応用の研究). 数理解析研究所講究録 2001, 1199: 220-227

ISSUE DATE:

2001-04

URL:

http://hdl.handle.net/2433/64911

数理解析研究所講究録
1199巻 2001年 220-227

整数行列のFrobenius標準形のモジュラー計算法*

図書館情報大学　森継　修一(Shuichi MORITSUGU) †
国立情報学研究所　栗山　和子(Kazuko KURIYAMA) ‡

1　はじめに

一般に線形代数の教科書では、行列の代表的な標準形としてJordan標準形が示されていることが多い。ただし、Jordan標準形の計算には、全固有値が属するまで体の代数拡大が必要であり、行列の要素が厳密表現の有理数で与えられている場合に対しても、Jordan標準形には一般に複素数が必要である。これを数式処理で実行する際には、固有値を代数的数として扱うため、拡大体上の計算の効率化に難点がある。また、「Jordanブロックの並び順を除いて」一意的という性質にも留意する必要がある。

これに対し、Frobenius標準形（有理標準形）は、体の拡大を必要とせず、行列要素間の四則演算（有理演算）のみで計算が可能であり、その結果はブロックの並び順まで含めて完全に一意的である。さらにFrobenius標準形は、もとの行列の特性多項式・最小多項式、固有値の代数的・幾何学的重複度や対応する（一般）固有ベクトルの構成などの完全な情報をJordan標準形と同等に含んでおり[14][16]、数式処理による行列計算のアルゴリズムを考えるにはJordan標準形よりもFrobenius標準形を基礎とする方が適している。

本研究の直接の動機は、固有値法による連立代数方程式の解法[22][17]において、実験の結果、全体の計算時間の中で有理数行列のFrobenius標準形の計算が支配的だったことである。Gröbner Basisの計算が飛躍的に進歩しているのに対して、固有値法全体を効率化するためには、行列計算の部分の改良が不可欠である。

2　行列のFrobenius標準形

以下の議論は任意の体の元を要素とする行列に対して成り立つが、具体的には、厳密な数値を表現する有理数にとるものとする。すなわち、$A=[a_{ij}]$, $a_{ij}\in\mathbf{Q}$ とおく。

*本研究は「平成12年度図書館情報大学特別研究(C)」の助成に基づく。
†moritsug@ulis.ac.jp
‡kuriyama@nii.ac.jp

定義 1 (コンパニオン行列) 次の $n \times n$ 正方行列

$$C = \begin{bmatrix} 0 & 1 & & & \\ 0 & 0 & \ddots & & O \\ \vdots & \vdots & \ddots & \ddots & \\ 0 & 0 & \cdots & 0 & 1 \\ c_0 & c_1 & \cdots & c_{n-2} & c_{n-1} \end{bmatrix} \tag{1}$$

は、多項式 $f(x) = x^n - c_{n-1}x^{n-1} - \cdots - c_1x - c_0$ に随伴するコンパニオン行列と呼ばれる。特に、1 次多項式 $f(x) = x - c_0$ のコンパニオン行列は 1×1 行列 $[c_0]$ とする。

補題 2 コンパニオン行列 C の特性多項式 $\varphi_C(x)$ と最小多項式 $\phi_C(x)$ は、随伴多項式 $f(x)$ に一致する。

定理 3 (Frobenius 標準形) 任意の $n \times n$ 正方行列 A は、適当な正則行列 S により次の形のブロック対角行列に相似変換され、これを A の Frobenius（または有理）標準形という。

$$F = S^{-1}AS = C_1 \oplus C_2 \oplus \cdots \oplus C_t. \tag{2}$$

各ブロック行列 C_i $(i = 1, \cdots, t)$ は、$m_i \times m_i$ コンパニオン行列 (1) であり、C_{i+1} の随伴多項式 $\varphi_{i+1}(x)$ は、C_i の随伴多項式 $\varphi_i(x)$ を割り切る $(i = 1, \ldots, t-1)$。与えられた行列 A に対して、(2) の形の行列は常に存在し、かつ一意的に決まる。さらに、A の最小多項式 $\phi_A(x)$ は $\varphi_1(x)$ に一致し、A の特性多項式 $\varphi_A(x)$ は $\varphi_1(x) \cdot \varphi_2(x) \cdots \varphi_t(x)$ で与えられる。

行列を (2) のようなブロック対角形に変換して特性多項式を求める方法が Danilevskii 法 [2][4] として古くから知られているが、そこでは $\varphi_t(x) \mid \varphi_{t-1}(x) \mid \cdots \mid \varphi_1(x)$ という整除条件を満たすことを要求していないので、結果が一意とは限らず、得られる $\varphi_1(x)$ は必ずしも A の最小多項式に一致しない。

これに対して本稿では、韓・伊理 [9] によるアルゴリズムに従い、定理 3 の厳密な意味での有理標準形 [5] を求める。その方法は、次の基本変形操作を組み合わせて、Gauss 消去法に類似の消去計算を実行するものである。

定義 4 (基本変形) 任意の $n \times n$ 正方行列 A に対する次の 3 つの操作を基本変形と呼ぶ。

$op1(k, \ell)$: A の第 k 行と第 ℓ 行を入れ換え、続いて第 k 列と第 ℓ 列を入れ換える。
$op2(k, c)$: A の第 k 行を c^{-1} 倍し、続いて第 k 列を c 倍する。
$op3(k, \ell, c)$: A の第 k 行に第 ℓ 行の c 倍を加え、続いて第 ℓ 列から第 k 列の c 倍を引く。

これらの基本変形は行列の相似変換 $(A \mapsto S^{-1}AS)$ として表現され、行に対する操作が左の変換行列 S^{-1} に、列に対する操作が右の変換行列 S に対応している。(ちなみに Gauss 消去法は、行に対する基本変形のみで実現されるので、$A \mapsto GA$ と表される。)

消去に必要な相似変換を $\cdots S_3^{-1}\left(S_2^{-1}\left(S_1^{-1}AS_1\right)S_2\right)S_3\cdots$ のように順次適用していくと、最終的に式 (2) における F, S, S^{-1} が得られるが、基本変形操作の定義より、この過程で必要となるのは有理数同士の四則演算のみであることに注意する。

注意 5 コンパニオン行列として (1) を転置させた形で定義し、Frobenius 標準形を (2) の転置で定義する流儀もある。実際、本稿におけるプログラムの本体も [9] に従って F^T を求めている。その場合は、

$$F^T = S^{-1}A^TS \qquad \Longleftrightarrow \qquad F = S^TAS^{-T}$$

という関係を利用し、A の代わりに A^T から計算を始めて、その結果を再度転置すればよい。この際、変換行列は、転置されるだけでなく、左右が入れ換わることに注意する。

注意 6 行列 A の要素をすべて整数にとった場合、その Frobenius 標準形の各要素も整数となる。A の特性多項式 $\varphi_A(x) = \det(xE - A)$ の係数は、行列式の計算規則から必ず整数になり、各コンパニオンブロック (1) の最下行の要素は、$\varphi_A(x)$ の因子の係数に対応するからである。ただし、変換行列 S, S^{-1} については、いずれか一方を整数行列にとることはできるが、現在のアルゴリズムでは両者を同時に整数行列にとることはできない。

3 モジュラー計算アルゴリズム

本稿では、さしあたって、整数行列 A のみを対象に考える。また、同時に計算する変換行列としては、固有ベクトルの計算への応用に必要な、$AS = SF$ をみたす S のみを求めることとし、S の各要素が整数になるようアルゴリズムを構成するものとする。

このように設定したとしても、基本変形 $op2(k, c)$ には「c^{-1} 倍する」という操作が含まれているため除算が必要であり、途中の段階では行列要素として有理数が現れることになる。筆者らは、この過程において有理数計算を避けると同時に要素の成長を最大限押さえる「分数なし計算法」[10][15] を以前に発表しているが、今回は、ここにモジュラー計算法の適用を試みる。基本変形における有理演算はすべて素数 p を法として実行可能なので、$\mathbf{Z}_p$ での Frobenius 標準形 $A_pS_p = S_pF_p$ が同じプログラムで求められる。(Euclid 互除法により、$cs + pt = 1$ をみたす s, t を求めれば、$s \equiv c^{-1} \pmod p$ となり、$op2$ の除算は乗算の形で実現される。)

注意 7 素数 p が unlucky な場合がある。すなわち、$\mathbf{Z}$ 上で求めた標準形 $AS = SF$ と $\mathbf{Z}_p$ での標準形 $A_pS_p = S_pF_p$ とにおいて、$F \not\equiv F_p \pmod p$ または $S \not\equiv S_p \pmod p$ となる場合が、(小さな p に対しては割合頻繁に) 起こりうる。これについては、消去計算の過程における pivot 選択の履歴から識別できるという指摘 [8] に基づき判定機能を実装したが、こ

の手法は、Gröbner Basis 計算にモジュラー法を適用し、S-Polynomial の頭項の履歴を用いて unlucky な素数を識別した [20] にも通じるものと思われる。ただし、素数が unlucky であるための理論的な十分条件は、現時点では未知である。

複数の素数を法として計算した結果 $(F_{p_1}, S_{p_1}), (F_{p_2}, S_{p_2}), \ldots$ から $\mathbf{Z}$ 上での F, S を復元するのに、本稿では Chinese Remainder Theorem の利用を考える。

定理 8 (CRT：法が 2 つの場合) m_1, m_2 が互いに素な整数のとき、連立合同式

$$\begin{cases} x \equiv a_1 & (\mathrm{mod}\ m_1) \\ x \equiv a_2 & (\mathrm{mod}\ m_2) \end{cases}$$

の解は、$m_1 s + m_2 t = 1$ を満たす s, t により、$x \equiv a_1 m_2 t + a_2 m_1 s \quad (\mathrm{mod}\ m_1 m_2)$ と表される。

したがって、素数 $p_1,\ p_2,\ p_3, \ldots$ に対して、法を $p_1,\ p_1 p_2,\ p_1 p_2 p_3, \ldots$ と上げていくには、

$$\begin{cases} x \equiv a_{k-1} & (\mathrm{mod}\ p_1 \cdots p_{k-1}) \\ x \equiv a_k & (\mathrm{mod}\ p_k) \end{cases} \qquad (k = 2,\ 3, \ldots)$$

に対して、定理 8 を繰り返し適用すればよい（Newton の補間公式）。行列 F, S に対しては、それぞれの要素 f_{ij}, s_{ij} について個別に（計 $2n^2$ 回）計算することになる。

一般に、変換行列 S の要素は、与えられた行列 A やその Frobenius 標準形 F の要素に比べて、遥かに巨大な整数となり、その上限の見積もりは（現在のところ）困難である。したがって、法をどこまで上げたらよいか事前には予想がつかないので、$\mathrm{mod}\ p_1 \cdots p_{k-1}$ での値 $S^{(k-1)}$ と $\mathrm{mod}\ p_1 \cdots p_{k-1} p_k$ での値 $S^{(k)}$ とが一致したら、整数上で $AS^{(k)} = S^{(k)} F^{(k)}$ を満たしているかどうかテストすることにより、終了判定とする。

以上を用いて、Frobenius 標準形のモジュラー計算をアルゴリズムとしてまとめると、以下のようになる。ここで、行列 A に対して $AS = SF$ なる S, F を求めるサブプログラムは、すでに出来ているものとする。

ただし、現在の実装では、unlucky な素数を識別するため、最初のいくつかの法に対して消去計算を実行してみて「正しい pivot 選択のパターン」を推測したうえで、以後、これと異なるパターンを与える法に基づく計算結果は CRT において除外することにしている。このとき最初の推測が間違っていると、このアルゴリズムは（用意した素数を使い尽くすまで）止まらなくなる。その意味で現時点では、（ある程度大きな素数を用いることにすれば失敗の可能性はかなり低いが）確率的アルゴリズムである。

アルゴリズム 9 (Frobenius 標準形のモジュラー計算)

% 入力：整数要素の $n \times n$ 行列 A　　ある程度大きな素数のリスト $\{p_1, p_2, \ldots, p_s\}$

表 14: CPU-Time(sec) for integer matrices I

n	#Prime	Modular	Fraction Free	Mod/FF
10	10	2.34	4.79	0.489
12	14	5.77	10.32	0.559
14	20	14.22	21.42	0.664
16	26	30.03	44.54	0.674
18	33	59.83	89.11	0.671
20	41	115.22	177.80	0.648
22	50	210.48	311.80	0.675
24	60	371.97	561.28	0.663

% 計算過程での $F^{(k)}, S^{(k)}$ は mod $p_1 \cdots p_k$ での値を表す。
% 出力：A の Frobenius 標準形 F と相似変換行列 S $(AS = SF)$
Compute F_{p_1}, S_{p_1} s.t. $A_{p_1} S_{p_1} \equiv S_{p_1} F_{p_1} \pmod{p_1}$;
$k := 1$; $F^{(1)} := F_{p_1}$; $S^{(1)} := S_{p_1}$;
loop : Do until $(S^{(k-1)} = S^{(k)})$
$k := k + 1$;
Compute F_{p_k}, S_{p_k} s.t. $A_{p_k} S_{p_k} \equiv S_{p_k} F_{p_k} \pmod{p_k}$;
Construct $F^{(k)}$ from $F^{(k-1)}$ and F_{p_k} by CRT;
Construct $S^{(k)}$ from $S^{(k-1)}$ and S_{p_k} by CRT;
If $(AS^{(k)} = S^{(k)}F^{(k)}$ (over $\mathbf{Z}))$ then return $\{F^{(k)}, S^{(k)}\}$ else goto loop;

4 実験的評価

以上に示したアルゴリズムを、数式処理システム Reduce3.6[7] および RLISP '88[12] を用いてインプリメントし、これまでのプログラムとの比較を行なった。機種は H9000VK270 (CPU : PA-8000, 160MHz)、メモリは 128MB で制限し、HP-UX 版 Reduce3.6 を使用した。

なお、素数のリストとしては、

$$\{p_1, p_2, \ldots, p_{500}\} = \{99999999977,\ 99999999947,\ \ldots,\ 99999987377\}$$

を用意し、すべての行列に対して、この順に適用した。

テスト用行列は 10×10 から 24×24 までの 8 個とし、その各要素は、Reduce の乱数発生関数を用いて、以下のように作成した。

表 15: CPU-Time(sec) for integer matrices II

n	#Prime	Modular	Fraction Free	Mod/FF
10	18	4.97	7.10	0.700
12	27	15.03	18.95	0.793
14	36	37.00	46.43	0.797
16	48	89.02	106.10	0.839
18	61	192.46	232.82	0.827
20	76	401.27	469.55	0.855
22	92	770.70	902.09	0.854
24	111	1466.18	1642.55	0.893

表 1 の行列　各要素は $-100 \sim 100$ の範囲の整数
表 2 の行列　各要素は $-10000 \sim 10000$ の範囲の整数

このようにランダムに行列要素を定めると、ほとんどの場合、コンパニオンブロック 1 つからなる Frobenius 標準形を持つ。標準形がブロックに分かれる場合には、単純に行列のサイズだけでは比較できなくなるので、今回の実験では、ブロック 1 つの標準形をもつ行列のみを対象とした。計算時間の測定結果を表 1,2 に示す。

今回のテスト用行列に対しては、unlucky な素数に当たることがなかったので、いずれも一度の計算で Frobenius 標準形 F と変換行列 S を求めることに成功している。計算時間としては、従来のプログラムと比べ、表 1 のもので約 65〜70%、表 2 のもので約 80〜90%となっている。

他の商業的数式処理システムでは、Maple[1] のみが Frobenius 標準形を計算する組込関数を持っていて、アルゴリズムは不明であるが、これはかなり高速で、整数行列に対しては上記のプログラムよりやや速い程度である。この Maple の関数の作成者自身が同アルゴリズムを Reduce 用に書き換えたパッケージもあるが、こちらは速度も遅く、メモリ消費も著しいため、実用には向かない。

5　今後の課題

整数行列に限ってのプログラムの基本形は動作させることができた（現在のところ、アルゴリズム的には、既知のものの組み合わせにすぎない）が、未解決の問題も多々残されている。

(1) 現在、変換行列 S を整数要素にとるようにしたため、CRT により、かなり大きな法

まで上げる必要が生じている。モジュラー計算からの有理数の再構成 [23][24] を組み込めば、S を有理数要素にとることも可能なので、どちらが効率的か調べる必要がある。

(2) 研究集会において、CRT にまつわる実装の高速化の技法をいくつかご教示いただいたので、それらも順次検討していきたい。

(3) 本来の目的である有理数要素の行列を扱えるようプログラムを拡張し、さらにアルゴリズムを評価・検討する必要がある。

行列の Frobenius 標準形をとりあげた研究は以前より多数発表されているが、その中では、本稿のような「消去法」によらないもの（むしろ計算量の議論が中心のもの）[3] [6] [11] [19] [21] が多い。オリジナルの Danilevskii 法に類似の方法に p-adic 計算を取り入れたものとしては [13][18] があるが、いずれも「標準形」までは求めず「ブロック対角化（あるいは上三角化）」に留めているので、特性多項式だけは求まるが、その他の応用には不十分と考えられる。今後、これらのアルゴリズムとの比較・再検討も必要である。

参 考 文 献

[1] Char, B. W., et al.: *Maple V Library Reference Manual*, Springer, N.Y., 1991.

[2] Danilevskii, A. M.: On the numerical solution of the secular equation, *Mat.Sb.*, **2**(1), 1937, 169–172. (in Russian).

[3] Eberly, W.: Black Box Frobenius Decompositions over Small Fields, *ISSAC 2000*, ACM, N.Y., 2000, 106–113.

[4] ファジェーエフ, ファジェーエバ: 線型代数の計算法 (上), 産業図書, 東京, 1970.

[5] Gantmacher, F. R.: *The Theory of Matrices (2nd ed.)*, **1**, Chelsea, N.Y., 1959.

[6] Giesbrecht, M.: Nearly Optimal Algorithms for Canonical Matrix Forms, *SIAM J. Comput.*, **24**(5), 1995, 948–969.

[7] Hearn, A. C.: *Reduce User's Manual (Ver. 3.6)*, RAND Corp., Santa Monica, 1995.

[8] Howell, J. A.: An Algorithm for the Exact Reduction of a Matrix to Frobenius Form Using Modular Arithmetic. I & II, *Math.Comp.*, **27**(124), 1973, 887–920.

[9] 韓太舜, 伊理正夫: ジョルダン標準形, 東京大学出版会, 東京, 1982.

[10] 栗山和子, 森継修一: 行列の有理標準形の分数なし計算法, 日本応用数理学会論文誌, **6**(3), 1996, 253–264.

[11] Lüneburg, H.: *On the Rational Normal Form of Endomorphisms: A Primer to Constructive Algebra*, Wissenschaftsverlag, Mannheim, 1987.

[12] Marti, J.: *RLISP '88: An Evolutionary Approach to Program Design and Reuse*, World Scientific, Singapore, 1993.

[13] Mathieu, M.-H. and Ford, D.: On p-adic Computation of the Rational Form of a Matrix, *J.Symbolic Computation*, **10**(5), 1990, 453–464.

[14] 森継修一, 栗山和子: 行列の Jordan 標準形の数式処理による厳密計算法, 日本応用数理学会論文誌, **2**(1), 1992, 91–103.

[15] Moritsugu, S. and Kuriyama, K.: Fraction-free Method for Computing Rational Normal Forms of Polynomial Matrices, RISC-Linz Report Series 97-18, RISC-Linz, 1997.

[16] Moritsugu, S. and Kuriyama, K.: On Multiple Zeros of Systems of Algebraic Equations, *ISSAC 99*, ACM, N.Y., 1999, 23–30.

[17] Moritsugu, S. and Kuriyama, K.: A Linear Algebra Method for Solving Systems of Algebraic Equations, *J.JSSAC*(日本数式処理学会誌), **7**(4), 2000, 2–22.

[18] Mukhopadhyay, A.: Exact Computation of the Characteristic Polynomial of an Integer Matrix, *AAECC 3*, *Lect. Notes in Comp. Sci.*, **229**, 1985, 316–324.

[19] Ozello, P.: *Calcul Exact des Formes de Jordan et de Frobenius d'une Matrice*, PhD thesis, Université Scientifique Technologique et Medicale de Grenoble, 1987.

[20] Sasaki, T. and Takeshima, T.: A Modular Method for Gröbner-basis Construction over **Q** and Solving System of Algebraic Equations, *J.Inf.Process.*, **12**(4), 1989, 371–379.

[21] Storjohann, A.: An $O(n^3)$ Algorithm for the Frobenius Normal Form, *ISSAC 98*, ACM, N.Y., 1998, 101–104.

[22] 竹島卓, 横山和弘: 連立代数方程式の一解法 - 剰余環上の線形写像の固有ベクトルの利用, 数式処理通信, **6**(4), 1990, 27–36.

[23] Wang, P. S.: A p-adic Algorithm for Univatiate Partial Fractions, *SYMSAC 81*, ACM, N.Y., 1981, 212–217.

[24] Wang, P. S., Guy, M. J. T., and Davenport, J. H.: P-adic Reconstruction of Rational Numbers, *ACM SIGSAM Bull.*, **16**(2), 1982, 2–3.